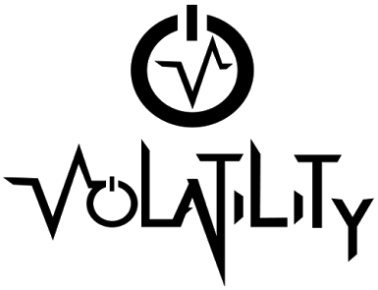
Installation / Resources



Check out the latest development build:
svn co
http://volatility.googlecode.com/svn/trunk/

Download a stable release:
https://code.google.com/p/volatility/downloads/list

Read full documentation:
https://code.google.com/p/volatility/wiki/VolatilityIntroduction?tm=6

Development Team Blog:
http://volatility-labs.blogspot.com

(Official) Training Contact:
voltraining@memoryanalysis.net

Follow: @volatility
Join: Volatility User's Mailing List

Basic Usage

Typical command components:
./vol.py -f [image] --profile=[profile] [plugin]

Display profiles, address spaces, plugins:
./vol.py --info

Display global command-line options:
./vol.py --help

Display plugin-specific arguments:
./vol.py [plugin] --help

Load plugins from an external directory:
./vol.py --plugins=[path] [plugin]

Specify a DTB or KDBG address:
./vol.py --dtb=[addr] --kdbg=[addr]

Specify an output file:
./vol.py --output-file=[file]

Image Identification

Get profile suggestions (OS and architecture):
imageinfo

Find and parse the debugger data block:
kdbgscan

Processes Listings

Basic active process listing:
pslist

Scan for hidden or terminated processes:
psscan

Cross reference processes with various lists:
psxview

Show processes in parent/child tree:
pstree

Process Information

Specify –O/--offset=OFFSET or -p/--pid 1 or –p/--pid 1,2,3 to any of these.

Display DLLs:
dlllist

Display details on VAD allocations:
vadinfo

Dump allocations to individual files:
vaddump --dump-dir=PATH

Dump all valid pages to a single file:
memdump --dump-dir=PATH

Display open handles:
handles
-t/--object-type=TYPE Mutant, File, Key, etc...
-s/--silent Hide unnamed handles

Display privileges:
privs
-r/--regex=REGEX Regex privilege name
-s/--silent Hide unnamed handles

Display SIDs:
getsids

Display environment variables:
envars

Display threads:
threads
-F/--filter=FILTER OrphanThread, HookedSSDT
-L/--listtags List filters

PE File Extraction

Specify -D/--dump-dir to any of these plugins to identify your desired output directory.

Dump a kernel module:
moddump
-r/--regex=REGEX Regex module name
-b/--base=BASE Module base address

Dump a process (without slack space):
procexedump

Dump a process (with slack space):
procmemdump

Dump DLLs in process memory:
dlldump
-r/--regex=REGEX Regex module name
-b/--base=BASE Module base address

Injected Code

Specify –O/--offset=OFFSET or -p/--pid 1 or –p/--pid 1,2,3 to any of these.

Find and extract injected code blocks:
malfind
-D/--dump-dir=PATH Dump findings here

Cross-reference DLLs with memory mapped files:
ldrmodules

Scan a block of code in process or kernel memory for imported APIs:
impscan
-p/--pid=PID Process ID
-b/--base=BASE Base address to scan
-s/--size=SIZE Size to scan from start of base

Logs / Histories

Recover event logs (XP/2003):
evtlogs
-S/--save-evt Save raw event logs
-D/--dump-dir=PATH Write to this directory

Recover command history:
cmdscan and consoles

Recover IE cache/Internet history:
iehistory
-L/--leak Find LEAK (deleted)
-R/--redr Find REDR (redirected)

Show running services:
svcscan
-v/--verbose Show ServiceDll from registry

Networking Information

Active info (XP/2003):
connections and sockets

Scan for residual info (XP/2003):
connscan and sockscan

Network info for Vista, 2008, and 7:
netscan

Kernel Memory

Display loaded kernel modules:
modules

Scan for hidden or residual modules:
modscan

Display recently unloaded modules:
unloadedmodules

Display timers and associated DPCs:
timers (x86 only)

Display kernel callbacks, notification routines:
callbacks (x86 only)

Audit the SSDT
ssdt
-v/--verbose Check for inline API hooks

Audit the IDT:
idt (x86 only)

Audit the GDT:
gdt (x86 only)

Audit driver dispatch (IRP) tables:
driverirp
-r/--regex=REGEX Regex driver name

Display device tree (find stacked drivers):
devicetree

Kernel Objects

Addresses shown as Offset(P) are physical locations in the memory dump file.

Scan for driver objects:
driverscan

Scan for mutexes:
mutantscan
-s/--silent Hide unnamed mutants

Scan for used/historical file objects:
filescan

Scan for symbolic link objects (shows drive mappings):
symlinkscan

Registry

Display cached hives:
hivelist

Print a key's values and data:
printkey
-o/--hive_offset=OFFSET Hive address (virtual)
-K/--key=KEY Key name

Dump userassist data:
userassist

Dump shellbags information:
shellbags

Dump the shimcache:
shimcache

Timelines

To create a timeline, tell volatility to create output in body file format. Combine the data and run sleuthkit's mactime to create a comma-separated values file.

timeliner --output=body > time.txt
shellbags --output-body >> time.txt
mftparser --output-body >> time.txt

mactime –b [time.txt] [-d] > csv.txt

Volshell

Interactive memory exploration:
volshell

List processes:
>>> ps()

Switch contexts by pid, offset, or name:
>>> cc(pid = 3028)

Acquire a process address space after using cc:
>>> process_space = self.proc.get_process_address_space()

Disassemble data in an address space
>>> dis(address, length, *space*)

Dump bytes, dwords or qwords:
>>> db(address, length, *space*)
>>> dd(address, length, *space*)
>>> dq(address, length, *space*)

Display a type/structure:
>>> dt("_EPROCESS")

Display a type/structure instance:
>>> dt("_EPROCESS", 0x820c92a0)

Create an object in kernel space:
>>> thread = obj.Object("_ETHREAD", offset = 0x820c92a0, vm = self.addrspace)

Dump Conversion

Create a raw memory dump from a hibernation, crash dump, firewire acquisition, virtualbox, vmware snapshot, hpak, or EWF file:
imagecopy –O/--output-image=FILE

Convert any of the aforementioned file types to a Windows crash dump compatible with Windbg:
raw2dmp –O/--output-image=FILE

API Hooks

Scan for API hooks:
apihooks (x86 only)
-R/--skip-kernel Don't check kernel modules
-P/--skip-process Don't check processes
-N/--no-whitelist Show all hooks (verbose)
-Q/--quick Scan faster

Yara Scanning

Scan for Yara signatures:
yarascan
-p/--pid=PID Process IDs to scan
-K/--kernel Scan kernel memory
-Y/--yara-rules=RULES String, regex, bytes, etc.
-y/--yara-file=FILE Yara rules file
-W/--wide Match Unicode strings

File System Resources

Find and parse MBR records in memory:
mbrparser
--hash=HASH Bootcode hash (up to RET)
--fullhash=HASH Bootcode hash (full)
--offset=OFFSET Offset (physical) to MBR
--check Check valid partition table
--disk=MBR MBR extracted from disk
--maxdistance=NUM Max Levenshtein distance

Scan for MFT records:
mftparser
-C/--check Hide entries w/ null timestamps
--output=body Output body format

Extract cached files (registry hives, executables):
dumpfiles
-S/--summary-file=FILE Summary file
-F/--filter=FILTER HandleTable, Vad, ...
-D/--dump-dir=PATH Output directory
-r/--regex=REGEX Regex filename to dump

GUI Memory

Sessions (shows RDP logins):
Sessions

Window stations (shows clipboard owners):
wndscan

Desktops (find ransomware):
deskscan

Display global and session atom tables:
atoms and atomscan

Dump the contents of the clipboard:
clipboard

Detect event hooks:
Eventhooks

Detect message hooks (keyloggers):
messagehooks

Dump the win32k.sys gahti:
gahti

Take a screen shot from the memory dump:
screenshot --dump-dir=PATH

Display visible and hidden windows:
windows and wintree

Display USER objects:
userhandles
-t/--type=TYPE TYPE_TIMER, TYPE_HWND, ...
-F/--free Show freed handles

Strings

Use GNU strings or Sysinternals strings.exe:
strings -a -td FILE > strings.txt
strings -a -td -el FILE >> strings.txt (Unicode)

strings.exe -q -o > strings.txt (Windows)

Translate the string addresses:
strings
-s/--string-file=FILE Input strings.txt file
-S/--scan

Password Recovery

Dump LSA secrets:
lsadump (x86 only)
-y/--sys-offset=OFFSET Offset to system hive
-s/--sec-offset=OFFSET Offset to security hive

Dump LM and NTLM hashes:
hashdump (x86 only)
-y/--sys-offset=OFFSET Offset to system hive
-s/--sam-offset=OFFSET

Extract RSA private keys and certificates:
dumpcerts
-s/--ssl Parse certificates with openssl (must be installed)